CONCEPTION AND EXPERIENCE OF METRICS-BASED SOFTWARE REUSE IN PRACTICE

Andreas Schmietendorf, Deutsche Telekom AG, Entwicklungszentrum Berlin

Evgeni Dimitrov, Deutsche Telekom AG, Entwicklungszentrum Berlin

Reiner Dumke, Otto-von-Guericke-Universität Magdeburg

Erik Foltin, Otto-von-Guericke-Universität Magdeburg

Michael Wipprecht, Deutsche Telekom AG, Entwicklungszentrum Berlin

ABSTRACT

There are already a number of studies and »success stories" about practical applications related to software reuse. For the most part however, the actual benefits of reuse, particularly for concrete technologies, are difficult to verify. The SW-WiVe project performed by Deutsche Telekom in collaboration with the Otto-von-Guericke University provides a detailed analysis and offers strategies for software reuse within industrial software development that can be subjected to critical evaluation. Traditional evaluation approaches, such as reuse metrics, were critically studied and necessary processes for continuous reuse were developed for this purpose. In a further step, currently available, valid reuse metrics for the software development process were classified and lacking metrics-based evaluation approaches were identified. This paper focuses on a description of the project's metrics-oriented terms of reference.

1 INTRODUCTION

In areas of industrial production such as the automotive industry, a high degree of reuse of previously manufactured intermediate products is considered to be a key for a high level of productivity, short product supply times, low costs per manufactured product, and a high level of quality. The basis for such a procedure is a process consisting of a division of labor between »suppliers" and »final assembly," as well as standards for functional and qualitative properties of the intermediate products used in order to fulfill customer requirements for the final product. Software developers also want to adopt this process, which has been successfully applied in all engineering-based industries, and are attempting in particular to use new technologies, such as object or component orientation, in order to improve what has been a mostly unsatisfactory situation up to the present.

The ability to achieve high reuse levels has been especially ascribed to object-oriented software development. As early as 1994, however, [Udell1994] proposed the thesis that the object-oriented approach would not be able to fulfill the high expectations for software reusability, and that software components would be more suitable for this. This discussion, which has been a matter of controversy up to the present, rejects other forms and types of reuse, as well as criteria other than software technology that influence reuse.

[Biggerstaff & Perlis 1989] provide the following generic definition of reuse:

»The reuse of software is the renewed use of artifacts and collected knowledge arising from the development of a software system when developing a new software system, in order to reduce the expenditure for creating and maintaining this new system."

Another definition for software reuse as a whole, i.e. for the reuse process, is provided by [Ezran 1998]:

»Software reuse is the systematic practice of developing software from a stock of building blocks, so that similarities in requirements and/or architecture between applications can be exploited to achieve substantial benefits in productivity, quality and business performance."

The following conclusions can be drawn from both definitions: Software reuse (referred to hereinafter simply as »reuse") must always be considered in relation to software development.

Artifacts and/or assets include a variety of reusable components, such as requirements, models, and implementation codes. Objectives for reuse include increased productivity, higher quality, cost reduction, lower maintenance costs, and also shorter development times. The characterization of reused components as a »stock of building blocks" and/or the two types of components relating to requirements or architecture (components) and the description of reuse as a »systematic" process.

2 EXAMPLES OF SUCCESSFUL REUSE

Capers Jones, an acknowledged expert in software metrics, recommends a reuse rate of more than 75% in order to achieve a »best in class" organization in software development [Jones 1998].

»Software reuse of design, code and test cases averages > 75 % for all projects"

Published examples of successful reuse can be found in [Jacobson 1997], with some companies in individual projects or application areas achieving a reuse level of almost 90%, according to their own figures:

- Within a period of ten years, beginning in 1984, Hewlett-Packard achieved a 25-50 percent reuse rate in firmware for instruments and printer products.
- AT&T: 40-92% in software for telecommunications support systems
- Brooklyn Union Gas: 90-95% at the process level and 67% at the level of user interfaces and business objects
- Ericsson AXE: 90% in hundreds of customer-specific configurations
- Motorola: 90% reuse and a tenfold increase in productivity in the development of compilers and compiler test tools.

General information on the essential advantages of reuse can also be found at the management level. Typical statements involve product supply times (time to market), which are reduced two to five times, error frequencies, which are reduced 5 to 10 times, and a five- to tenfold reduction in production costs.

None of these »success stories" provide a basis for their calculations, meaning that their statements can generally be dismissed as simply marketing. According to [Poulin 1997], there is no uniform model that establishes what counts as »reuse" and shows how to calculate the added value created in this way. This requirement means that the use of metrics can play a decisive role in successful reuse.

3 OVERVIEW OF REUSE ASPECTS

The »Fifth International Conference on Software Reuse 1998" classified the problem of reuse according to various aspects, and presented it schematically in a so-called »reuse diamond." A short description of the equally ranked aspects contained in this reuse diamond is presented below. The purpose of the description is to clearly demonstrate the complexity of the topic, as well as to present the content limits defined in the SW-WiVe project.

Strategy and Management - This aspect involves such elements as the establishment of reuse strategies and management support, as well as the need for an organizational development (frequently described as a reuse-centered organization).

People - The literature contains various role models [Jacobson 1997], [Sodhi 1998], [Coulange 1998] that interact closely with the selected organizational form.

Assets - The term »assets" describes all reusable products such as components, objects, requirements, analysis and design models, codes, documentation, etc. The term »artifacts" is often used as a synonym for »assets."

Technology - This refers primarily to software production environments and the repositories used to store necessary information on software development, as well as appropriate component databases containing reusable assets.

Process - The traditional software development process provides insufficient support for reuse. Thus necessary roles and/or personnel requirements, for example, are not defined.

Measurement - The use of software metrics serves primarily to provide an element of quantification in the entire reuse process. We will

go into greater detail on this problem in a separate section.

Because the applicability of metrics is closely associated with process quality, which is also expressed, for example, within the CMM model for evaluating the maturity of software development, the SW-WiVe project focuses on the aspects of »Process" and »Measurement."

4 Analysis of Available Software Metrics in Reuse

4.1 Analysis Procedure

The analysis of the initial situation is based upon a general framework proposed by the Otto-von-Guericke University at Magdeburg for the implementation of so-called »metrics programs» [Dumke 1998] and the classification developed by [Fenton 1991] using product, process, and resource as starting points for software measurement.

The framework mentioned above consists of a CAME[1] strategy, a CAME[2] framework, and the CAME[3] tools for the recording and processing of metrics. While the CAME strategy refers to aspects such as the need for the existence of a group to promote the implementation of software metrics and the need for management decisions, the CAME framework relates, for example, to the selection process of metrics, the analysis of scale properties of these metrics, the degree of coverage reached, and the efficiency of metrics usage by means of tools support. Because Deutsche Telekom has already made strategic decisions regarding the use of software metrics, the analysis concentrates on the selection of metrics (CAME framework) and the determination of an overall degree of coverage for available reuse metrics.

4.2 Metrics Selection (Choice)

When selecting metrics, the traditional approach in evaluating software reuse has usually been based on »local optimization," in that

a an individual aspect (usually a cost aspect) is studied closely and used for evaluation, or

b a special (empirically established) target area is selected, and then questions for improvement are derived for this area and the metrics are determined in the sense of a GQM (Goal Question Metrics) approach according to [Basili et al. 1998].

[Poulin 1997] gives ***Relative Cost of Reuse (RCR)*** for simple components and ***Relative Cost of Writing for Reuse (RCWR)*** as examples of a) in the sense of special cost metrics which take product architecture into account to a certain degree.

[Ezran 1998] gives the following example for b), presented as an excerpt. In this example, the necessary metrics are determined almost directly based upon the objectives, although the metrics are only generally stated and are not subjected to any theoretical measurement analysis.

[1] C - Community, A - Acceptance, M - Motivation, E - Engagement
[2] C - Choice, A - Adjustment, M - Migration, E - Efficiency
[3] C - Computer, A - Assisted, M - Measurement, E - Evaluation

Table 1: Examples of metrics given by [Ezran 1998]

Objectives	Repository use	Reuse expansion in a project	Component costs	Process costs, including reuse	Component quality	Reusability
Metrics	• Number of components • Number of accesses	• Reuse level per project • Number of accesses to the components per project	• RCR • RCWR	• Company ROI • Payment threshold values	• Reliability • Compre-hensibility	• Portability • Functional complete-ness

Further analysis of the literature available within the SW-WiVe project also showed in particular a lack of reuse metrics for product usage, the role of development personnel, and the influence of hardware. There is also a lack of metrics to consider the complexity effects of reuse based on the product, the development process, and the resources used.

4.3 Determining the Scale Properties (Adjustment)

In the second step, namely the determination of measurement properties (adjustment), it was found that the metrics approaches known from the literature primarily relate to the (reuse) cost share; in other words, they are »potentially ratio scaled" from an empirical standpoint. This therefore involves an indirect measurement or evaluation into which the properties of the development components cannot be incorporated. Poulin proposed the following empirical valuation-models [Poulin 1997]:

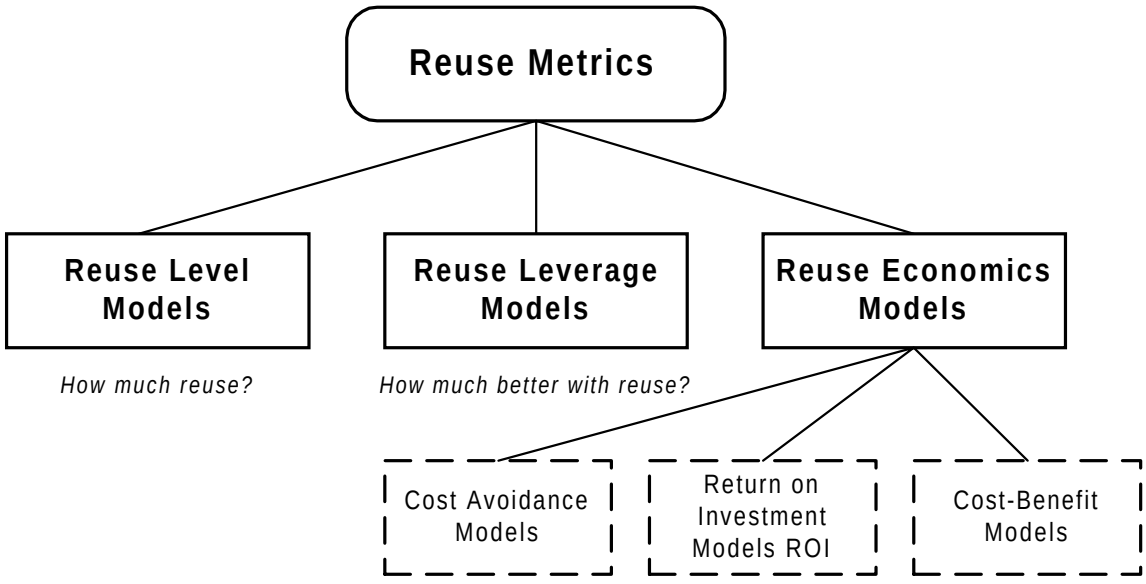

Fig. 1: Reuse metrics and economic models

A study of the formulaic presentation of metrics listed in [Barnard 1998], [Poulin 1997], and [Ezran 1998] showed that the currently proposed reuse metrics generally do not involve any dimensions.[4] Altogether we find following metrics-situation regarding their measurement properties:

[4] We define a (software) dimension as a relatively (ratio) scaled metric provided with a measurement unit.

Metric	Measure-ing type	Calculation	Scale Properties
Amortization (Gaffney)	Calculation	(RCR + RCWR/n –1) * R +1, n – number of expected reuse, R – proportion of reused code in the product	potentially ratio scaled
COCOMO Modification (Balda)	Estimation respectively Measure-ment	$LM = a_i N_i^b$ N_1= KDSI for unique code developed N_2= KDSI for code developed for reuse N_3= KDSI from reused code N_4= KDSI from modified components	potentially ratio scaled
Consumer/Producer Reuse (Chen)	Estimation	Reuse(S,P) = Σ weights S – Systems, P – Repositories, AppReusePerc(S) = Reuse(S) / Size(S) RepReusePerc(S,p) =Reuse(S,p)/Size(p)	ordinal scaled
Cost Benefit (Bollinger)	Estimation	Benefit = withoutReuse – withReuse - ReuseInvestment	potentially ratio scaled
Cost Benefit (Henders.)	Estimation	ROI= (withoutReuse – withReuse)/ producedReuseComponents	potentially ratio scaled
Cost Benefit (Malan)	Estimation	Costs= (withoutReuse – withReuse) - ReuseOverhead (considering a set of products)	potentially ratio scaled
Cost Benefits (Poulin)	Estimation	Σ costs - Σ benefits	potentially ratio scaled
DISA (DoD)	Calculation	Defense Information Systems Agency: - lines of new custom code - lines of new reusable code (code written for reuse by others) - lines of verbatim reused code - lines of adapted reused code	potentially ratio scaled
Generic Reuse (Bieman)	Calculation	Number of generic classes or functions or parameters (for templates)	potentially ratio scaled
RCA (Poulin)	Estimation	ReuseCostAvoidance=Development CostAvoidance+ServiceCostAvoidance	ordinal scaled
RCR (Poulin)	Estimation	Portion of effort that it takes a similar component without modification (black-box reuse)	potentially ratio scaled
RCWR (Poulin)	Estimation	Portion of effort that is take to write a similar reusable component	potentially ratio scaled
Reusable Index (Sodhi)	Valuation	Component: 4 – most reusable, . . ., 1 – least reusable	ordinal scaled
Reusability (Barnard)	Calculation	Calculation-formula based on CBO, DIT, NFC and other OO-metrics	ordinal scaled
Reuse Level (Frakes)	Calculation	(InternalReuseLevel + ExternalReuse Level)/ TotalNumberOfItems	ordinal scaled
Reuse Leverage (Banker)	Calculation	TotalObjectsUsed/ NewObjectsBuilt	ordinal scaled
Reuse Leverage (Poulin)	Estimation	ProductivityWithReuse/ ProductivityWithoutReuse	potentially ratio scaled
Reuse Percent (Poulin)	Estimation	ReusedSoftware/TotalSoftware	ordinal scaled
ROI (Poulin)	Estimation	ReturnOnInvestment = Σ Coeff * Reuse in Phase i	potentially ratio scaled
RVA (Poulin)	Calculation	ReuseValueAdded= (TotalSource Statements + SourceInstructionsReused ByOthers)/ (TotalSourceStatements – ReusedSourceInstruction)	ordinal scaled
Verbatim Reuse (Bieman)	Calculation	Classes (functions) included by use or has relationships	potentially ratio scaled

That shows, that the most cost-benefit-metrics ('without Reuse', 'with Reuse') only a "proclamatic" character have. At the scale properties the best case was accepted presently. For the potentially ratio scaled was accepted therefore, that an empirical experience to the possible statement of a measurement unit is given.

4.4 Support of a Metrics-Based Workflow (Migration)

Regarding the migration following starting-points are recognizable:

- the reuse-process is among other things supported by metrics, that a assessment is viewed to the reusability of existing software-components. The so-called reusability-metrics become classifies (to [Poulin 1997]) into empirical and qualitative methods, the are presently on the other hand module- or has component-oriented. Examples of such metrics are from Selby, Caldiera/Basili, and Torres for empirical methods and the IBM Reusability Guidelines respectively the so-called 3C-Modells for qualitative methods.

- for a reusable-component as product-component we can find the special problem of the contemplation-area (especially regarding the internal and external reuse), the so-called Boundary-problem.

The following can be stated in regard to migration, i.e. the applicability of reuse metrics in all phases of the software life-cycle and all created products:

- The course of development is generally considered only cumulatively in regard to the respective reuse costs or benefits.

- The progress of the process is considered only in terms of how the components' reusability relates to the actual reuse.

- A special process consideration ultimately serves to determine the degree to which components in repositories are reused.

- Reused components are considered only in regard to program lines, objects or classes, and so-called »items" (program excerpts or functions).

Only Bieman and Barnard go out from a detailed model of object-oriented systems, look at however not the development-phases. On the other hand is showy, that exclusively reusable code comes in into the analyses and other development documents is viewed doesn't quantify. On the empirical side the essential goals are described with metrics (expenses, benefits, expenditure etc.), however followed the registration of concrete measurement values likely only over coarse time-estimations.

4.5 Efficiency of the Reuse Process (Efficiency)

In existing solutions, the efficiency of a metrics-based reuse process refers primarily to the (computer-aided) access statistics. Otherwise, the manual collection of reuse figures dominates, although the (statistical) analysis of these figures is generally also made with the help of a computer. Bieman, for example, created a component-based, tool-supported metrics analysis.

Metrics databases were generally not used. Metrics data are usually organized only implicitly in the repositories in regard to access statistics.

5 Requirements of a Metrics-Supported Reuse Approach

Based on the analyses performed and the general metrics program framework developed by the Otto-von-Guericke University at Magdeburg, the following general requirements were established for a software reuse that is quantified and thus can be evaluated:

1. In software reuse, the already existing reuse metrics can also be used, as well as the metrics valid for other paradigms, in order to make them easier to quantify and thus evaluate.

2. Selected metrics should be relatively scaled to the extent possible.

    - An empirical experience, which can service the basis for an initial scaling unit, must be prepared or selected for component-based metrics,

- A component relationship that makes a detailed evaluation possible must be created for empirical metrics (costs, expenditure, etc.).

3. Above all, the selection of metrics must include every area (i.e. the products, the process, and the resources) in the evaluation.

4. The objective for the selected metrics must also be to achieve the highest possible level of automation. This also includes »traditional" measurement tools. The following table describes the current situation:

Table 2: Reuse metrics and tool-support

Metric	for Paradigm	Tool-support	Empirical experiences
Amortization (Gaffney)	no restriction	no	Case study
COCOMO Modification (Balda)	no restriction	no	Calculation-example
Consumer/Producer Reuse (Chen)	no restriction	AST	Case study (C-Programs)
Cost Benefit (Bollinger)	*not practical*		
Cost Benefit (Henders.)	*not practical*		
Cost Benefit (Malan)	*not practical*		
Cost Benefits (Poulin)	no restriction	no	
DISA (DoD)	no restriction	no statement	Calculation-example
Generic Reuse (Bieman)	OOSE	ARMA-Tool for Ada	Case study
RCA (Poulin)	no restriction	Worksheet defined	
RCR (Poulin)	Module oriented	no	Literature-comparison (∅ = 0.2)
RCWR (Poulin)	Module oriented	no	Literature-comparison (∅ = 2)
Reusable Index (Sodhi)	no restriction	no	no
Reusability (Barnard)	OOSE	MOOD	experiment-covered
Reuse Level (Frakes)	imperative	rl tool	Case study (C-Programs)
Reuse Leverage (Banker)	OOSE	no	Case study
Reuse Leverage (Poulin)	*not practical*		
Reuse Percent (Poulin)	no restriction	no	Calculation-example
ROI (Poulin)	sequence life cycle	Worksheet defined	Calculation-example
RVA (Poulin)	imperative	Worksheet defined	Calculation-example
Verbatim Reuse (Bieman)	OOSE	ARMA-Tool for Ada	Case study

5. The measured values themselves must be integrated into metrics databases. This also applies for access statistics of individual reused components.

6. In addition to the predominately cost-oriented metrics, the aspects of quality improvement or influence must also be considered. That is valid for such questions how

   - the previous determination the quality of imported software-components,

   - the "updating" of the quality at the utilization of components with a definite quality of Q1 in a system with a quality of Q2.

7. Particular attention must be paid to the motivation for reuse. For this is to investigate

   - how far developers early to have included,

   - a reuse bonus model could be introduced,

   - effective experiments the basis could form.

An efficient use of metrics is however of the analysis of the current software development situation and the so recognizable success-promises starting-points dependent.

6 SUGGESTION OF A METRICS-BASED REUSE-CONCEPT

6.1 Integration of the software-development- and reuse-process

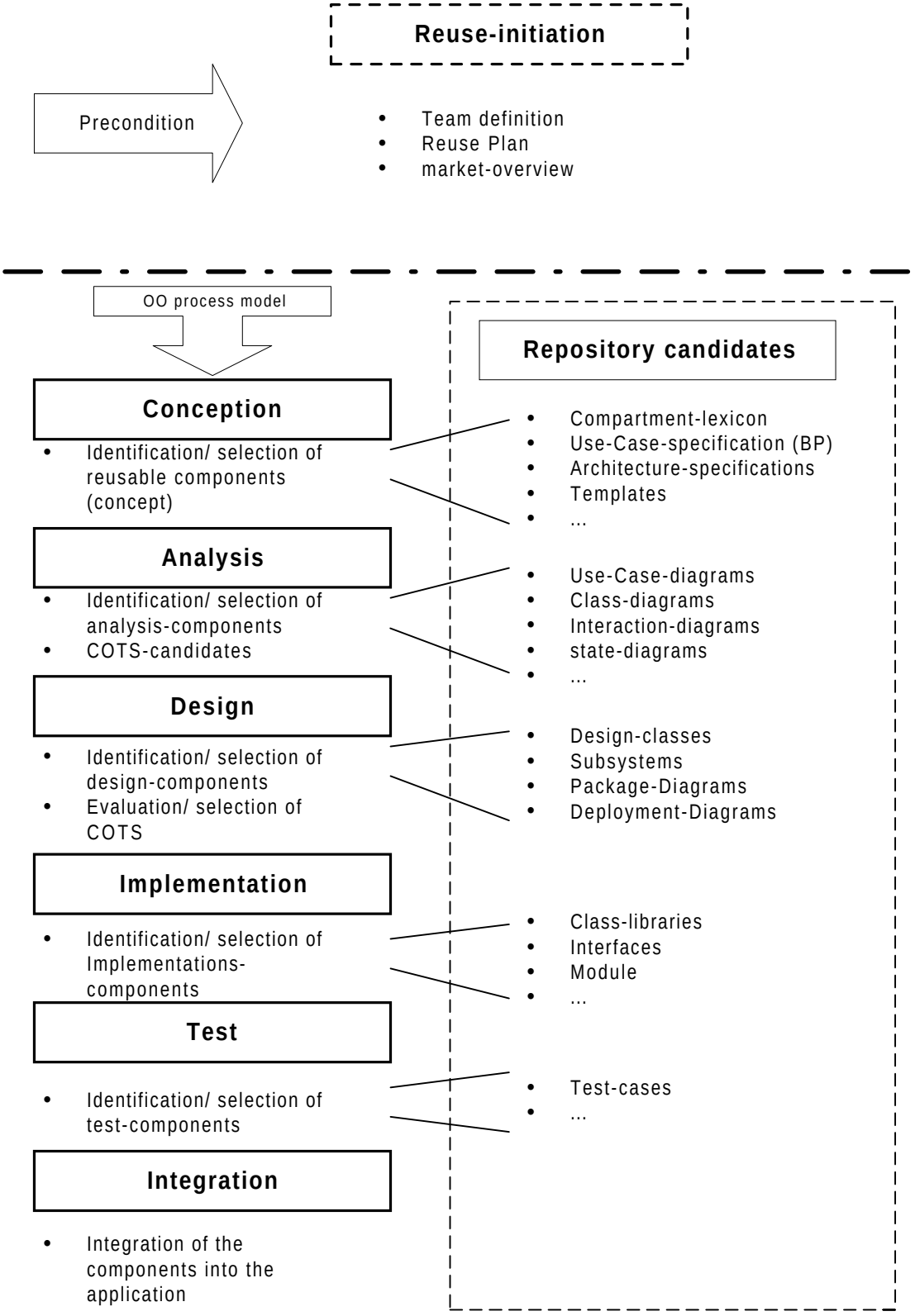

Fig. 2: Integration of the software-development- and reuse-process

6.2 Evaluating Reuse Maturity

Sodhi's Software Reuse Maturity Model

Using the »Capability Maturity Model (CMM)" proposed by the SEI[5], which evaluates the maturity of a software development in five levels, [Sodhi 1998] proposed a corresponding »Software Reuse Maturity Model." The starting point for considerations was the insufficient evaluation of necessary details in the process for software reuse within the CMM. This evaluation model contains the levels described briefly below. The achievement of a level is determined by means of a questionnaire.

Level 1 - Informal practices: Describes the entry level of reuse. Practical forms of reuse are utilized, although the organization does not explicitly support the process. The type and manner of reuse basically depends upon the personal involvement of the individual developers.

Level 2 - Formal software reuse: If Level 2 is achieved, then reproducible results of reuse already exist through a certain formalization of the process. For example, quality assurance for the reused components has been introduced, time and cost estimates take reuse into account to a certain extent, and initial steps have been taken to define the necessary processes for reuse.

Level 3 - Implementation of formal reuse: A mandatory process has been defined for reuse. Repositories are now available with certified and sufficiently tested reusable components. Coordinated technical application domains offer a basis for the reuse of architectural components. A set of initial metrics must be introduced in this phase, referring in particular to the testing or clarification of advantages from reuse.

Level 4 - Management return on investment ROI[6]: The metrics introduced in Level 3 are used, for example, to provide evidence for a ROI or to present practical benefits of implemented component databases by means of reuse indices for each component. The expenditures necessary for reuse can be quantified in regard to both already implemented application domains as well as new ones.

Level 5 - Optimization: All processes for software reuse are optimized. An implemented procedure supports the economically optimal achievement of a higher level for software reuse in each case. The effects of a necessary »reengineering" of existing assets can be tracked, and rules also exist for measuring the aggressive development of implemented reuse processes.

Adjusted Variants for Evaluating Reuse Maturity

In summarizing the evaluation possibilities studied, the SW-WiVe project proposed a variant for evaluating software reuse maturity that is adapted to Deutsche Telekom's requirements. This variant uses the following three levels:

Reuse initiatives: This level basically implies the contents of Level 2, and considers the aspects of reuse separately from one another for the most part. Typical here are the implementation of prototypical reuse databases with more technical assets, but without defining the processes necessary for effective use and without a strategic decision by management on a procedure for reuse.

Assessment-based reuse: This level basically implies the contents of Level 3. The introduction of valid metrics related to reuse means that processes, resources, and the product (assets) can be evaluated. The prototypical reuse databases that were implemented are developed further into comprehensive approaches, and are thus available to everyone involved in the software development. A reuse-centered organizational development has been established in order to develop domain-specific components, which are already recognized as standard.

Controlled reuse Process: This level basically implies the contents of Level 4, i.e. a metrics-supported reuse approach is established in order to clearly quantify the value added by reuse. In addition, the necessary organizational development has progressed to the point that

[5] Software Engineering Institute
[6] ROI Return on Investment

centers exist for actual component creation, management of available components, and actual application development.

6.3 Metrics in the process of the software-reuse

Selection of the metrics and statement of the attributes (Choice/Adjustment) corresponding the GQM-method (Goal Question Metric). With it a definition of the goals is presupposed.

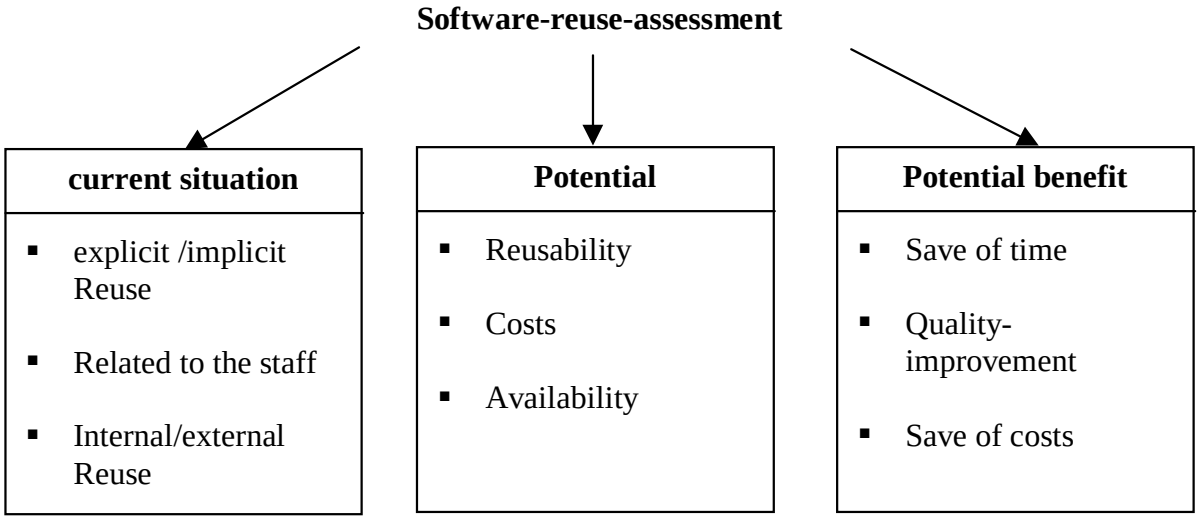

Fig. 3: Initial stage for the metrics choice

The goals already imply the questions and led immediate to the metrics. The concrete alignment of the metrics is however regarding product, processes or resources to plan. We want to restrict ourselves first of all on the **product**.

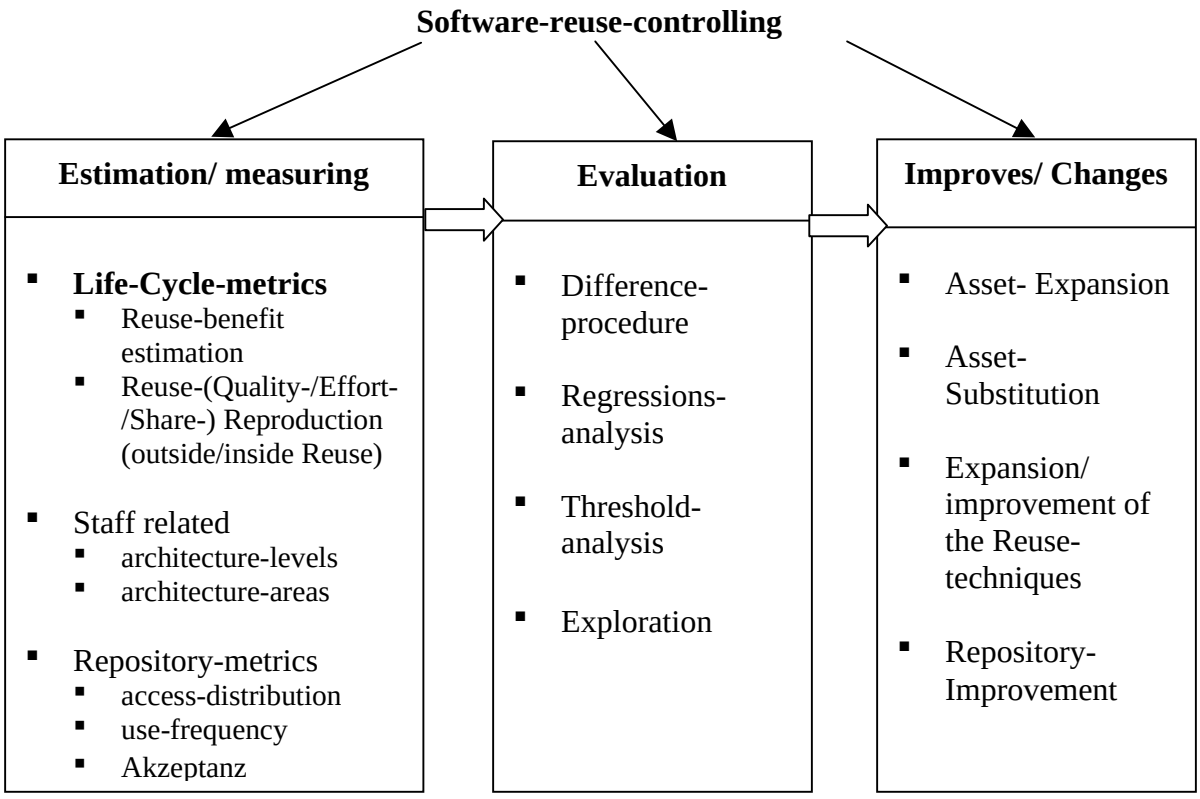

Fig. 4: Supplement of the metrics with life-cycle- and architecture-assessments

7 SUMMARY AND PENDING TOPICS

In conclusion, it is clear that the level of knowledge regarding software reuse depends upon the respective (process) level. In the same way that relationships (quality, productivity, efficiency, etc.) in software development have until now largely been observed only on a »case-by-case" basis or studied very generally and without differentiation, there is also a lack of comprehensive or complex studies of the reuse process. All phases of development lack applicable reuse metrics to consider the various aspects of reuse, particularly in relation to a metrics-supported reuse approach.

The general relationships between reuse models and economical models, which are ultimately essential for evaluating reuse, have not yet been subjected to sufficiently comprehensive analysis, nor have they been standardized independently of individual companies.

The currently practical process consists of preparing a company-specific evaluation model for reuse, in order to at least achieve an appropriate level of transparency within the company. This naturally means that comparisons cannot be made with other companies.

The measurement approaches presented in the literature (particularly by [Poulin 1997]) have not yet been validated to the extent that they can be used to create solutions or reuse strategies that can be sufficiently generalized. Current solutions incorporate almost exclusively process experiences in software development. An essential basis for validation is the use of a metrics database to manage the various series of measurements and achieve the necessary statistical reliability.

In order to expand the range of experience for reuse technologies, practical applications are needed for other application classes and areas of application. This applies above all for differently »scaled" companies and IT departments.

In addition to improving reuse technologies, the expenditures and the costs of reuse must themselves be examined. For example, it is also

necessary to study the extent to which the reuse technologies utilized in each case may increase the complexity of the development to an unacceptable degree and ultimately make it unmanageable.

The SW-WiVe project paid particular attention to the specified success factors for software reuse, and shows how one part of the problem can be approached and/or how a solution could be created. In addition to the studies presented here regarding the use of metrics, the project also undertook the company-specific definition of necessary processes for software reuse, as well as a proposal for a generic description of assets and an initial approach for formalizing this description through the use of metrics. The project also made a practical study of reused components already employed by specific companies, and identified features that are closely associated with a successful reuse of these components.

8 SOURCES

[Barnard 1998] Barnard, J.: A new reusability metric for object-oriented software. Software Quality Journal, 7 (1998), S. 35 – 50.

[Basili et al. 1988] Basili, V.; Rombach, D.: The TAME-Project: Towards Improvement-Oriented Software Engineering. IEEE Trans. on Software Engineering, 14(1988)6, S. 758 – 773.

[Biggerstaff & Perlis 1989] Biggerstaff, T. and Perlis, A.: Software Reusability, Volume I and II in the Frontier Series. ACM Press, 1989.

[Coulange 1998] CoulangeB.: Software Reuse, Springer-Verlag, London 1998.

[Dumke 1998] Dumke, R., Foltin, E., Winkler, A.S.: A Framework for Object-Oriented Software Measurement and Evaluation. Proceedings of the IASTED Conference on Software Engineering, Oct. 28-31, 1998, Las Vegas, S. 129 – 132.

[Ezran 1998] Ezran, M., Morisio, M., Tully, C.: Practical Software Reuse: The essential Guide. Paris: Freelife Publ., 1998.

[Fenton 1991]Fenton, N.: Software Metrics - A Rigorous Approach. London: Chapmann & Hall Inc., 1991.

[Jacobson 1997] Jacobson, I, Griss, M., Jonsson, P.: Software Reuse (Architecture, Process and Organization for Business Success), Reading/MA: Addison-Wesley 1997.

[Jones 1998] Jones, C.: What it means to be »Best in class" for Software. Vers. 5, Software Productivity Research, 1998.

[Poulin 1997] Poulin, J.S.: Measuring Software Reuse. Principles, Practices, and Economic Models. Reading/MA: Addison-Wesley 1997.

[Sodhi 1998] Sodhi, J.; Sodhi, P.: Software Reuse. Domain Analysis and Design Processes. New York ...: McGraw-Hill 1998.

[Udell 1994] Udell, J. Component software, BYTE, 19(5):46-55, 1994.